**RAP Green Field – Managed**

Application keeps the alerts from the system in a centralize app. There one can reprocess the issue that happens, delete, and edit alerts:



Transparent/Persistent Tables:

|  |
| --- |
| @EndUserText.label : 'Alerts Yard Logistics'  @AbapCatalog.enhancement.category : #NOT\_EXTENSIBLE  @AbapCatalog.tableCategory : #TRANSPARENT  @AbapCatalog.deliveryClass : #A  @AbapCatalog.dataMaintenance : #ALLOWED  **define** **table** zyl\_alerts **{**  **key** mandt **:** mandt **not** **null;**  **key** alert\_guid **:** sysuuid\_c32 **not** **null;**  **include** zyl\_alerts\_s\_without\_key**;**  **}** |

|  |
| --- |
| @EndUserText.label : 'Draft table for entity ZYL\_I\_ALERTS'  @AbapCatalog.enhancement.category : #EXTENSIBLE\_ANY  @AbapCatalog.tableCategory : #TRANSPARENT  @AbapCatalog.deliveryClass : #A  @AbapCatalog.dataMaintenance : #RESTRICTED  **define** **table** zyl\_alerts\_draft **{**  **key** mandt **:** mandt **not** **null;**  **key** alertguid **:** sysuuid\_c32 **not** **null;**  yardno **:** /sapyl/e\_yard\_no**;**  alertsource **:** zyl\_alert\_source\_e**;**  alertstatus **:** zyl\_alert\_status\_e**;**  userreserved **:** usnam**;**  funcname **:** funcname**;**  reftype **:** zyl\_alert\_reftype\_e**;**  reference **:** zyl\_alert\_reference\_e**;**  alertcomment **:** zyl\_comment\_e**;**  reprocesscounter **:** zyl\_reprocess\_counter**;**  createdby **:** /bofu/user\_id\_created\_by**;**  createdon **:** /bobf/creationdatetime**;**  changedby **:** /bofu/user\_id\_changed\_by**;**  changedon **:** /bobf/lastchangedatetime**;**  "%admin" **:** **include** sych\_bdl\_draft\_admin\_inc**;**  **}** |

|  |
| --- |
| @EndUserText.label : 'Alert Messages'  @AbapCatalog.enhancement.category : #NOT\_CLASSIFIED  @AbapCatalog.tableCategory : #TRANSPARENT  @AbapCatalog.deliveryClass : #A  @AbapCatalog.dataMaintenance : #RESTRICTED  **define** **table** zyl\_alerts\_msgs **{**  **key** mandt **:** mandt **not** **null;**  **key** alert\_guid **:** sysuuid\_c32 **not** **null;**  **key** msg\_counter **:** /cfg/de\_message\_counter **not** **null;**  msg **:** natxt**;**  **}** |

|  |
| --- |
| @EndUserText.label : 'Draft table for entity ZYL\_I\_ALERT\_MESSAGES'  @AbapCatalog.enhancement.category : #EXTENSIBLE\_ANY  @AbapCatalog.tableCategory : #TRANSPARENT  @AbapCatalog.deliveryClass : #A  @AbapCatalog.dataMaintenance : #RESTRICTED  **define** **table** zyl\_alerts\_msgsd **{**  **key** mandt **:** mandt **not** **null;**  **key** alertguid **:** sysuuid\_c32 **not** **null;**  **key** msgcounter **:** /cfg/de\_message\_counter **not** **null;**  msg **:** natxt**;**  "%admin" **:** **include** sych\_bdl\_draft\_admin\_inc**;**  **}** |

CDS Views

|  |
| --- |
| @AbapCatalog.viewEnhancementCategory: [#NONE]  @AccessControl.authorizationCheck: #NOT\_REQUIRED  @EndUserText.label: 'Interface View for Alerts'  @Metadata.ignorePropagatedAnnotations: true  @ObjectModel.usageType:{  serviceQuality: #A,  sizeCategory: #L,  dataClass: #TRANSACTIONAL  }  **define** **root** **view** **entity** ZYL\_I\_ALERTS  **as** **select** **from** zyl\_alerts  **composition** **[**0**..\*]** **of** ZYL\_I\_ALERT\_MESSAGES **as** \_AlertMsgs //on $projection.AlertGuid = \_AlertMsgs.AlertGuid  **association** **[**0**..**1**]** **to** ZYL\_I\_ALERT\_SOURCE **as** \_AlertSource **on** **$projection.**AlertSource **=** \_AlertSource**.**Value  **association** **[**0**..**1**]** **to** ZYL\_I\_ALERT\_STATUS **as** \_AlertStatus **on** **$projection.**AlertStatus **=** \_AlertStatus**.**Value  **association** **[**0**..**1**]** **to** ZYL\_SCMB\_ESDUS **as** \_Param **on** **$projection.**YardNo **=** \_Param**.**YardNo  **{**  **key** alert\_guid **as** AlertGuid**,**  yard\_no **as** YardNo**,**  alert\_source **as** AlertSource**,**  alert\_status **as** AlertStatus**,**  user\_reserved **as** UserReserved**,**  funcname **as** Funcname**,**  reftype **as** Reftype**,**  reference **as** Reference**,**  alert\_comment **as** AlertComment**,**  reprocess\_counter **as** ReprocessCounter**,**  created\_by **as** CreatedBy**,**  created\_on **as** CreatedOn**,**  changed\_by **as** ChangedBy**,**  changed\_on **as** ChangedOn**,**  /\* Associations \*/  \_AlertMsgs**,**  \_AlertSource**,**  \_AlertStatus**,**  \_Param  **}** |

|  |
| --- |
| @AbapCatalog.viewEnhancementCategory: [#NONE]  @AccessControl.authorizationCheck: #NOT\_REQUIRED  @EndUserText.label: 'Interface View for Alert Messages'  @Metadata.ignorePropagatedAnnotations: true  @ObjectModel.usageType:{  serviceQuality: #A,  sizeCategory: #L,  dataClass: #TRANSACTIONAL  }  **define** **view** **entity** ZYL\_I\_ALERT\_MESSAGES  **as** **select** **from** zyl\_alerts\_msgs  **association** **to** **parent** ZYL\_I\_ALERTS **as** \_Alert **on** **$projection.**AlertGuid **=** \_Alert**.**AlertGuid  **{**  **key** alert\_guid **as** AlertGuid**,**  **key** msg\_counter **as** MsgCounter**,**  @Semantics.text: true  msg **as** Msg**,**  \_Alert  **}** |

|  |
| --- |
| @AbapCatalog.viewEnhancementCategory: [#NONE]  @AccessControl.authorizationCheck: #NOT\_REQUIRED  @EndUserText.label: 'Alert Source Values(Domain)'  @Metadata.ignorePropagatedAnnotations: true  @ObjectModel.usageType:{  serviceQuality: #X,  sizeCategory: #S,  dataClass: #MIXED  }  @ObjectModel.resultSet: {  sizeCategory: #XS //DropDown  }  **define** **view** **entity** ZYL\_I\_ALERT\_SOURCE  **as** **select** **from** dd07t  **{**  //key valpos as ValPos,  **key** domvalue\_l **as** Value**,**  @Semantics.text: true  ddtext **as** Text  **}**  **where**  domname **=** 'ZYL\_ALERT\_SOURCE\_D'  **and** ddlanguage **=** **$session.system\_language** |

|  |
| --- |
| @AbapCatalog.viewEnhancementCategory: [#NONE]  @AccessControl.authorizationCheck: #NOT\_REQUIRED  @EndUserText.label: 'Alert Source Values(Domain)'  @Metadata.ignorePropagatedAnnotations: true  @ObjectModel.usageType:{  serviceQuality: #X,  sizeCategory: #S,  dataClass: #MIXED  }  @ObjectModel.resultSet: {  sizeCategory: #XS //DropDown  }  **define** **view** **entity** ZYL\_I\_ALERT\_STATUS  **as** **select** **from** dd07t  **{**  **key** domvalue\_l **as** Value**,**  @Semantics.text: true  ddtext **as** Text  **}**  **where**  domname **=** 'ZYL\_ALERT\_STATUS\_D'  **and** ddlanguage **=** **$session.system\_language** |

|  |
| --- |
| @AbapCatalog.viewEnhancementCategory: [#NONE]  @AccessControl.authorizationCheck: #NOT\_REQUIRED  @EndUserText.label: 'CDS for /SCMB/ESDUS YardNo'  @Metadata.ignorePropagatedAnnotations: true  @ObjectModel.usageType:{  serviceQuality: #X,  sizeCategory: #S,  dataClass: #MIXED  }  @ObjectModel.resultSet: {  sizeCategory: #XS  }  **define** **view** **entity** ZYL\_SCMB\_ESDUS  **as** **select** **from** /scmb/esdus  **{**  **key** uname **as** Uname**,**  **key** action **as** Action**,**  **key** element **as** Element**,**  active **as** YardNo  **}**  **where**  uname **=** **$session.user**  **and** action **=** '/SAPYL/DEFAULT'  **and** element **=** 'YARD\_NO' |

|  |
| --- |
| @AbapCatalog.viewEnhancementCategory: [#NONE]  @AccessControl.authorizationCheck: #NOT\_REQUIRED  @EndUserText.label: 'Consumption View for Alerts Messages'  @Search.searchable: false  @Metadata.allowExtensions: true  **define** **view** **entity** ZYL\_C\_ALERTS\_MESSAGES  // provider contract transactional\_query  **as** **projection** **on** ZYL\_I\_ALERT\_MESSAGES  **{**  **key** AlertGuid**,**  **key** MsgCounter**,**  Msg**,**  /\* Associations \*/  \_Alert **:** **redirected** **to** **parent** ZYL\_C\_ALERTS  **}** |

|  |
| --- |
| @AccessControl.authorizationCheck: #NOT\_REQUIRED  @Search.searchable: false  @Metadata.allowExtensions: true  @EndUserText.label: 'Consumption View for Alerts'  **define** **root** **view** **entity** ZYL\_C\_ALERTS  **provider** **contract** **transactional\_query**  **as** **projection** **on** ZYL\_I\_ALERTS  // Association goes on root view of the projection view  //association [0..1] to ZYL\_I\_ALERT\_STATUS as \_AlertStatus on $projection.AlertStatus = \_AlertStatus.Value  //association [0..1] to ZYL\_I\_ALERT\_SOURCE as \_AlertSource on $projection.AlertSource = \_AlertSource.Value  **{**  **key** AlertGuid**,**  YardNo**,**  @ObjectModel.text.element: ['AlertSourceText']  AlertSource**,**  \_AlertSource**.**Text **as** AlertSourceText**,**  @ObjectModel.text.element: ['AlertStatusText']  AlertStatus**,**  \_AlertStatus**.**Text **as** AlertStatusText**,**  UserReserved**,**  Funcname**,**  Reftype**,**  Reference**,**  AlertComment**,**  ReprocessCounter**,**  CreatedBy**,**  CreatedOn**,**  ChangedBy**,**  ChangedOn**,**  \_Param**.**YardNo **as** ParamYardNo**,**  /\* Associations \*/  \_AlertMsgs**:** **redirected** **to** **composition** **child** ZYL\_C\_ALERTS\_MESSAGES**,**  \_AlertSource**,**  \_AlertStatus**,**  \_Param  **}** |

|  |
| --- |
| @AbapCatalog.viewEnhancementCategory: [#NONE]  @AccessControl.authorizationCheck: #NOT\_REQUIRED  @EndUserText.label: 'Alerts Counter for local Yard'  @Metadata.ignorePropagatedAnnotations: true  @ObjectModel.usageType:{  serviceQuality: #X,  sizeCategory: #S,  dataClass: #MIXED  }  **define** **root** **view** **entity** ZYL\_C\_ALERTS\_COUNTER  **as** **select** **from** zyl\_alerts **as** alerts  **inner** **join** ZYL\_SCMB\_ESDUS **as** \_Param **on** alerts**.**yard\_no **=** \_Param**.**YardNo  **{**  **key** alerts**.**alert\_guid **as** AlertGuid**,**  alerts**.**yard\_no **as** YardNo  **}**  **where**  \_Param**.**Uname **=** **$session.user**  // + IDoc Alerts  **union** **select** **from** zyl\_alerts  **{**  **key** alert\_guid **as** AlertGuid**,**  yard\_no **as** YardNo  **}**  **where**  yard\_no **=** '' |

|  |
| --- |
| @AccessControl.authorizationCheck: #NOT\_REQUIRED  @Search.searchable: false  @Metadata.allowExtensions: true  @EndUserText.label: 'Local Yard saved Parameter'  **define** **root** **view** **entity** ZYL\_C\_LOCAL\_YARD  **provider** **contract** **transactional\_query**  **as** **projection** **on** ZYL\_I\_LOCAL\_YARD  **{**  **key** Uname**,**  **key** Action**,**  **key** Element**,**  YardNo  **}** |

|  |
| --- |
| @AbapCatalog.viewEnhancementCategory: [#NONE]  @AccessControl.authorizationCheck: #NOT\_REQUIRED  @EndUserText.label: 'Interface View for Local Yard'  @Metadata.ignorePropagatedAnnotations: true  @ObjectModel.usageType:{  serviceQuality: #A,  sizeCategory: #L,  dataClass: #TRANSACTIONAL  }  **define** **root** **view** **entity** ZYL\_I\_LOCAL\_YARD  **as** **select** **from** /scmb/esdus  **{**  **key** uname **as** Uname**,**  **key** action **as** Action**,**  **key** element **as** Element**,**  active **as** YardNo  **}**  **where**  uname **=** **$session.user**  **and** action **=** '/SAPYL/DEFAULT'  **and** element **=** 'YARD\_NO' |

**Behavior Projection**

|  |
| --- |
| **projection;**  //strict;  **use** **draft;**  **define** **behavior** **for** ZYL\_C\_ALERTS //alias <alias\_name>  **{**  //use create;  **use** **update;**  **use** **delete;**  **use** **action** reprocess**;**  **use** **association** \_AlertMsgs **{** **with** **draft;** **}**  **}**  **define** **behavior** **for** ZYL\_C\_ALERTS\_MESSAGES **use** **etag**  **{**  **use** **association** \_Alert **{** **with** **draft;** **}**  // use update;  // use delete;  **}** |

**Behavior Definitions**

|  |
| --- |
| **managed** **implementation** **in** **class** zbp\_yl\_i\_alerts **unique;**  //strict;  **with** **draft;**  **define** **behavior** **for** ZYL\_I\_ALERTS //alias Alerts  **persistent** **table** zyl\_alerts  **draft** **table** zyl\_alerts\_draft  //with unmanaged save //if unmanaged data needs to be updated manually  **lock** **master**  **total** **etag** AlertGuid //-> only for drafts  **authorization** **master** **(** **global,** **instance** **)**  **etag** **master** AlertGuid  **{**  **field** **(** **readonly** **)** AlertGuid**,** YardNo**,** AlertSource**,** AlertStatus**,** Funcname**,** Reftype**,** Reference**,** ReprocessCounter**,** CreatedBy**,** CreatedOn**,** ChangedBy**,** ChangedOn**;**  //create;  **update;** //( features : instance );  **delete;** //( features : instance );  // association \_AlertMsgs { with draft; }  //field (mandatory)  **association** \_AlertMsgs **{** **create;** **with** **draft;** **}**  **action** **(** **features** **:** **instance** **)** reprocess **result** **[0..\*]** **$self;**  // action set\_status\_booked result [1] $self;  **mapping** **for** ZYL\_ALERTS  **{**  AlertGuid **=** alert\_guid**;**  YardNo **=** yard\_no**;**  AlertSource **=** alert\_source**;**  AlertStatus **=** alert\_status**;**  UserReserved **=** user\_reserved**;**  Funcname **=** funcname**;**  Reftype **=** reftype**;**  Reference **=** reference**;**  AlertComment **=** alert\_comment**;**  ReprocessCounter **=** reprocess\_counter**;**  CreatedBy **=** created\_by**;**  CreatedOn **=** created\_on**;**  ChangedBy **=** changed\_by**;**  ChangedOn **=** changed\_on**;**  **}**  **}**  **define** **behavior** **for** ZYL\_I\_ALERT\_MESSAGES **alias** AlertMessages  **persistent** **table** zyl\_alerts\_msgs  **draft** **table** zyl\_alerts\_msgsd  **lock** **dependent** **by** \_Alert  **authorization** **dependent** **by** \_Alert  **etag** **master** AlertGuid  **{**  // update;  // delete;  **association** \_Alert **{** **with** **draft;** **}**  **field** **(** **readonly** **)** AlertGuid**,** MsgCounter**,** Msg**;**  **mapping** **for** zyl\_alerts\_msgs  **{**  AlertGuid **=** alert\_guid**;**  MsgCounter **=** msg\_counter**;**  Msg **=** msg**;**  **}**  **}** |

**Service Definition**

|  |
| --- |
| @EndUserText.label: 'Service Definition for Alerts'  **define** **service** ZYL\_SD\_ALERTS **{**  **expose** ZYL\_C\_ALERTS**;**  **expose** ZYL\_C\_LOCAL\_YARD**;**  **expose** ZYL\_C\_ALERTS\_COUNTER**;**  **expose** ZYL\_C\_ALERTS\_MESSAGES**;**  **}** |

**Service Binding**

**A screenshot of a computer

Description automatically generated**

**Classes**

|  |
| --- |
| zbp\_yl\_i\_alerts |
| CLASS zbp\_yl\_i\_alerts DEFINITION PUBLIC ABSTRACT FINAL FOR BEHAVIOR OF zyl\_i\_alerts.  ENDCLASS.  CLASS zbp\_yl\_i\_alerts IMPLEMENTATION.  ENDCLASS. |
| \*CLASS lhc\_alertmessages DEFINITION INHERITING FROM cl\_abap\_behavior\_handler.  \*  \* PRIVATE SECTION.  \*  \* METHODS get\_global\_features FOR GLOBAL FEATURES  \* IMPORTING REQUEST requested\_features FOR AlertMessages RESULT result.  \*  \*ENDCLASS.  \*  \*CLASS lhc\_alertmessages IMPLEMENTATION.  \*  \* METHOD get\_global\_features.  \* ENDMETHOD.  \*  \*ENDCLASS.  CLASS lhc\_zyl\_i\_alerts DEFINITION INHERITING FROM cl\_abap\_behavior\_handler.  PRIVATE SECTION.  METHODS get\_global\_authorizations FOR GLOBAL AUTHORIZATION  IMPORTING REQUEST requested\_authorizations FOR zyl\_i\_alerts RESULT result.  METHODS get\_instance\_authorizations FOR INSTANCE AUTHORIZATION  IMPORTING keys REQUEST requested\_authorizations FOR zyl\_i\_alerts RESULT result.  METHODS get\_instance\_features FOR INSTANCE FEATURES  IMPORTING keys REQUEST requested\_features FOR zyl\_i\_alerts RESULT result.  METHODS reprocess FOR MODIFY  IMPORTING it\_key\_for\_action FOR ACTION zyl\_i\_alerts~reprocess REQUEST it\_requested\_fields RESULT et\_action\_result.  ENDCLASS.  CLASS lhc\_zyl\_i\_alerts IMPLEMENTATION.  \* METHOD modify.  \*  \* ENDMETHOD.  METHOD get\_global\_authorizations.  \* DATA(lo\_alerts) = NEW zyl\_cl\_alerting( ).  \* AUTHORITY-CHECK OBJECT '/SAPYL/YO'  \* ID 'ACTVT' FIELD '16'  \* ID '/SAPYL/YN' FIELD <fs\_cds>-yardno  \* ID '/SAPYL/YOT' FIELD <fs\_yos>-ordertype  \* ID 'BO\_SERVICE' FIELD 'ACT\_ROOT\_CHECKOUT'.  \* IF sy-subrc <> 0.  \* result-%action-reprocess = if\_abap\_behv=>auth-unauthorized.  \* result-%action-edit = if\_abap\_behv=>auth-unauthorized.  \* result-%delete = if\_abap\_behv=>auth-unauthorized.  \* ELSE.  \* result-%action-reprocess = if\_abap\_behv=>auth-allowed.  \* result-%action-edit = if\_abap\_behv=>auth-allowed.  \* result-%delete = if\_abap\_behv=>auth-allowed.  \* ENDIF.  \* APPEND VALUE #( %msg = lo\_alerts->map\_bapi\_to\_behv\_message( is\_message\_bapi = VALUE #( type = 'E' message = 'Test' ) )  \* %global = if\_abap\_behv=>mk-on  \* ) TO reported-zyl\_i\_alerts.  ENDMETHOD.  METHOD get\_instance\_authorizations.  ENDMETHOD.  METHOD get\_instance\_features.  " Read the active flag of the existing members  READ ENTITIES OF zyl\_i\_alerts IN LOCAL MODE  ENTITY zyl\_i\_alerts  FIELDS ( alertguid alertstatus ) WITH CORRESPONDING #( keys )  RESULT DATA(lt\_result\_data)  FAILED failed.  result = VALUE #( FOR ls\_result\_data IN lt\_result\_data  LET status = COND #( WHEN ls\_result\_data-alertstatus = '5' "Completed  THEN if\_abap\_behv=>fc-o-disabled  ELSE if\_abap\_behv=>fc-o-enabled )  IN  ( %tky = ls\_result\_data-%tky  %action-reprocess = status  %action-edit = status  %features-%action-reprocess = status  %features-%action-edit = status  ) ).  ENDMETHOD.  METHOD reprocess.  DATA(lo\_alerts) = NEW zyl\_cl\_alerting( ).  LOOP AT it\_key\_for\_action ASSIGNING FIELD-SYMBOL(<fs\_key>).  TRY.  DATA(lt\_return) = lo\_alerts->zyl\_if\_alerting~reprocess\_alert( iv\_alert\_guid = <fs\_key>-alertguid ).  IF lt\_return IS INITIAL.  APPEND VALUE #( type = 'S' id = 'ZYL\_ALERTING' number = 001 message\_v1 = <fs\_key>-alertguid ) TO lt\_return.  ENDIF.  LOOP AT lt\_return ASSIGNING FIELD-SYMBOL(<fs\_return>).  APPEND VALUE #(  %cid = <fs\_key>-%cid\_ref  alertguid = <fs\_key>-alertguid  %msg = lo\_alerts->map\_bapi\_to\_behv\_message( is\_message\_bapi = <fs\_return> )  \* %element-alertguid = if\_abap\_behv=>mk-on  \* %action-reprocess = if\_abap\_behv=>mk-on  \* %element-field2 = if\_abap\_behv=>mk-on  \* %element-\_association = if\_abap\_behv=>mk-on  \* %state\_area = 'state\_area'  \* %path = VALUE #( <root>-%is\_draft = <child>-%is\_draft  \* <root>-<key> = <child>-<parent\_key\_in\_child\_entity> )  ) TO reported-zyl\_i\_alerts.  ENDLOOP.  CATCH zyl\_cx\_alerting\_error.  ENDTRY.  ENDLOOP.  ENDMETHOD.  ENDCLASS.  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  \*  \* Saver class implements the save sequence for data persistence  \*  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  CLASS lsc\_saver DEFINITION INHERITING FROM cl\_abap\_behavior\_saver.  PROTECTED SECTION.  \* METHODS finalize REDEFINITION.  \* METHODS check\_before\_save REDEFINITION.  \* METHODS save REDEFINITION.  \* METHODS cleanup REDEFINITION.  \* METHODS save\_modified REDEFINITION.  ENDCLASS.  CLASS lsc\_saver IMPLEMENTATION.  \* METHOD finalize.  \* ENDMETHOD.  \* METHOD check\_before\_save.  \* ENDMETHOD.  \* METHOD save.  \* ENDMETHOD.  \* METHOD cleanup.  \* ENDMETHOD.  \* METHOD save\_modified.  \* ENDMETHOD.  ENDCLASS. |

|  |
| --- |
| ZYL\_CL\_ALERTING |
| class ZYL\_CL\_ALERTING definition  public  final  create public .  public section.  interfaces ZYL\_IF\_ALERTING .  interfaces IF\_RAP\_PLMI\_BAPI\_MSG\_CONVERT .  aliases ALERT\_SOURCE\_APOMON  for ZYL\_IF\_ALERTING~ALERT\_SOURCE\_APOMON .  aliases ALERT\_SOURCE\_CUSTOM  for ZYL\_IF\_ALERTING~ALERT\_SOURCE\_CUSTOM .  aliases ALERT\_SOURCE\_IDOC  for ZYL\_IF\_ALERTING~ALERT\_SOURCE\_IDOC .  aliases ALERT\_FROM\_CUSTOM  for ZYL\_IF\_ALERTING~ALERT\_FROM\_CUSTOM .  aliases CREATE\_ALERT\_FROM\_APOMON  for ZYL\_IF\_ALERTING~CREATE\_ALERT\_FROM\_APOMON .  aliases CREATE\_ALERT\_FROM\_IDOC  for ZYL\_IF\_ALERTING~CREATE\_ALERT\_FROM\_IDOC .  aliases CREATE\_ALERT\_MESSAGES  for ZYL\_IF\_ALERTING~CREATE\_ALERT\_MESSAGES .  aliases SEND\_ALERT  for ZYL\_IF\_ALERTING~SEND\_ALERT .  aliases UPDATE\_ALERT  for ZYL\_IF\_ALERTING~UPDATE\_ALERT .  types:  BEGIN OF ty\_status\_error,  status TYPE teds1-status,  END OF ty\_status\_error .  types:  ty\_r\_status\_error TYPE RANGE OF ty\_status\_error .  methods GET\_IDOC\_STATUS\_ERROR  returning  value(RR\_STATUS\_ERROR) type TY\_R\_STATUS\_ERROR .  methods MAP\_BAPI\_TO\_BEHV\_MESSAGE  importing  !IS\_MESSAGE\_BAPI type BAPIRET2  returning  value(RO\_MESSAGE\_BEHV) type ref to IF\_ABAP\_BEHV\_MESSAGE .  methods GET\_SLG1\_MESSAGES  importing  !IV\_OBJECT type BALOBJ\_D optional  !IV\_SUBOBJECT type BALSUBOBJ optional  !IV\_EXT\_IDENTIF type BALNREXT optional  returning  value(ET\_BAPIRET2) type BAPIRET2\_T .  PROTECTED SECTION.  PRIVATE SECTION.  TYPES:  ty\_lt\_existing\_alert TYPE STANDARD TABLE OF zyl\_alerts WITH DEFAULT KEY .  DATA alert\_from\_db TYPE zyl\_alerts\_s .  DATA alert\_from\_db\_full TYPE zyl\_alerts .  DATA alert\_parameters\_from\_db TYPE zyl\_alerts\_param\_t .  DATA to\_update\_alert TYPE zyl\_alerts\_s .  DATA to\_update\_alert\_full TYPE zyl\_alerts .  DATA alert\_guid TYPE sysuuid\_c32 .  DATA to\_update\_alert\_parameters TYPE zyl\_alerts\_param\_t .  METHODS check\_alert\_exists  IMPORTING  !is\_alert TYPE zyl\_alerts\_s\_without\_key  RETURNING  VALUE(rv\_exists) TYPE flag.  METHODS create\_alert  RAISING  zyl\_cx\_alerting\_error .  METHODS create\_parameters\_for\_alert  RAISING  zyl\_cx\_alerting\_error .  METHODS enqueue\_alert  RAISING  zyl\_cx\_alerting\_error .  METHODS dequeue\_alert .  METHODS get\_alert .  METHODS get\_alert\_parameters .  METHODS get\_existing\_open\_alert  IMPORTING  !is\_alert TYPE zyl\_alerts\_s\_without\_key  RETURNING  VALUE(rt\_existing\_alert) TYPE ty\_lt\_existing\_alert .  ENDCLASS.  CLASS ZYL\_CL\_ALERTING IMPLEMENTATION.  METHOD check\_alert\_exists.  IF is\_alert-reference IS NOT INITIAL.  SELECT SINGLE \*  FROM zyl\_alerts  INTO @DATA(lv\_reference)  WHERE reference = @is\_alert-reference.  IF sy-subrc EQ 0.  rv\_exists = abap\_true.  ENDIF.  ENDIF.  ENDMETHOD.  METHOD zyl\_if\_alerting~create\_alert\_from\_apomon.  MOVE-CORRESPONDING is\_alert TO alert\_from\_db .  alert\_from\_db-alert\_source = zyl\_if\_alerting=>alert\_source\_apomon.  ENDMETHOD.  METHOD zyl\_if\_alerting~create\_alert\_from\_idoc.  IF NOT check\_alert\_exists( is\_alert = is\_alert ).  MOVE-CORRESPONDING is\_alert TO to\_update\_alert .  to\_update\_alert\_parameters = it\_alert\_parameter.  to\_update\_alert-alert\_source = zyl\_if\_alerting=>alert\_source\_idoc.  TRY.  create\_alert( ).  CATCH zyl\_cx\_alerting\_error.  RAISE EXCEPTION TYPE zyl\_cx\_alerting\_error  EXPORTING  textid = zyl\_cx\_alerting\_error=>create\_alert\_insert\_duplicate  iv\_tabname = 'ZYL\_ALERTS'.  ENDTRY.  IF alert\_guid IS NOT INITIAL.  TRY.  create\_parameters\_for\_alert( ).  CATCH zyl\_cx\_alerting\_error.  RAISE EXCEPTION TYPE zyl\_cx\_alerting\_error  EXPORTING  textid = zyl\_cx\_alerting\_error=>create\_alert\_insert\_duplicate  iv\_tabname = 'ZYL\_ALERTS\_PARAM'.  ENDTRY.  ENDIF.  rv\_alert\_guid = alert\_guid.  "Add Messages from IDOC  DATA lv\_idoc TYPE edids-docnum.  DATA lt\_idocs\_log TYPE edids\_tt.  SPLIT is\_alert-reference AT '/' INTO TABLE DATA(lt\_split).  lv\_idoc = VALUE #( lt\_split[ 1 ] OPTIONAL ).  SELECT \* FROM edids INTO TABLE lt\_idocs\_log  WHERE docnum EQ lv\_idoc.  IF sy-subrc = 0.  SORT lt\_idocs\_log DESCENDING BY docnum logdat logtim.  DELETE ADJACENT DUPLICATES FROM lt\_idocs\_log COMPARING docnum.  ENDIF.  CHECK NOT lt\_idocs\_log IS INITIAL.  LOOP AT lt\_idocs\_log INTO DATA(ls\_idoc\_log).  DATA(lv\_tabix) = sy-tabix.  PERFORM prepare\_st\_error\_text IN PROGRAM rbdprocess CHANGING ls\_idoc\_log IF FOUND.  MODIFY lt\_idocs\_log FROM ls\_idoc\_log INDEX lv\_tabix TRANSPORTING statxt.  ENDLOOP.  create\_alert\_messages( iv\_alert = alert\_guid  it\_idoc\_msgs = lt\_idocs\_log  it\_bapiret2 = me->get\_slg1\_messages( iv\_ext\_identif = CONV #( lv\_idoc ) ) ).  ENDIF.  ENDMETHOD.  METHOD zyl\_if\_alerting~send\_alert.  ENDMETHOD.  METHOD zyl\_if\_alerting~update\_alert.  IF is\_alert\_update IS SUPPLIED.  alert\_guid = is\_alert\_update-alert\_guid.  get\_alert( ).  CLEAR to\_update\_alert.  to\_update\_alert = is\_alert\_update.  ENDIF.  IF alert\_from\_db\_full IS NOT INITIAL.  DATA ls\_alert\_update TYPE zyl\_alerts.  MOVE-CORRESPONDING to\_update\_alert TO ls\_alert\_update.  "1. Enqueue  enqueue\_alert( ).  "2. Update  ls\_alert\_update-changed\_by = sy-uname.  GET TIME STAMP FIELD ls\_alert\_update-changed\_on.  UPDATE zyl\_alerts FROM ls\_alert\_update.  ENDIF.  "3. Dequeue  dequeue\_alert( ).  ENDMETHOD.  METHOD create\_alert.  "1. Assign GUID  TRY.  to\_update\_alert-alert\_guid = cl\_system\_uuid=>create\_uuid\_c32\_static( ).  CATCH cx\_uuid\_error.  RETURN.  ENDTRY.  alert\_guid = to\_update\_alert-alert\_guid.  to\_update\_alert-alert\_status = '1'.  to\_update\_alert-created\_by = sy-uname.  GET TIME STAMP FIELD to\_update\_alert-created\_on.  DATA: ls\_alerts TYPE zyl\_alerts.  MOVE-CORRESPONDING to\_update\_alert TO ls\_alerts.  INSERT zyl\_alerts FROM ls\_alerts.  IF sy-subrc <> 0.  RAISE EXCEPTION TYPE zyl\_cx\_alerting\_error  EXPORTING  textid = zyl\_cx\_alerting\_error=>create\_alert\_insert\_duplicate  iv\_tabname = 'ZYL\_ALERTS'.  ENDIF.  ENDMETHOD.  METHOD create\_parameters\_for\_alert.  LOOP AT to\_update\_alert\_parameters ASSIGNING FIELD-SYMBOL(<ls\_parameter>).  <ls\_parameter>-alert\_guid = alert\_guid.  IF <ls\_parameter>-para\_name IS INITIAL.  DELETE to\_update\_alert\_parameters INDEX sy-tabix.  ENDIF.  ENDLOOP.  DATA: lt\_alert\_parameter TYPE TABLE OF zyl\_alerts\_param.  MOVE-CORRESPONDING to\_update\_alert\_parameters TO lt\_alert\_parameter.  INSERT zyl\_alerts\_param FROM TABLE lt\_alert\_parameter.  IF sy-subrc <> 0.  RAISE EXCEPTION TYPE zyl\_cx\_alerting\_error  EXPORTING  textid = zyl\_cx\_alerting\_error=>create\_alert\_insert\_duplicate  iv\_tabname = 'ZYL\_ALERTS\_PARAM'.  ENDIF.  ENDMETHOD.  METHOD enqueue\_alert.  CALL FUNCTION 'ENQUEUE\_EZYL\_ALERTS'  EXPORTING  mode\_zyl\_alerts = 'X'  mandt = sy-mandt  alert\_guid = alert\_guid  x\_alert\_guid = space  \_scope = '2'  \_wait = space  \_collect = ' '  EXCEPTIONS  foreign\_lock = 1  system\_failure = 2  OTHERS = 3.  IF sy-subrc <> 0.  RAISE EXCEPTION TYPE zyl\_cx\_alerting\_error  EXPORTING  textid = zyl\_cx\_alerting\_error=>enqueue\_failed  iv\_guid = to\_update\_alert-alert\_guid.  ENDIF.  ENDMETHOD.  METHOD dequeue\_alert.  CALL FUNCTION 'DEQUEUE\_EZYL\_ALERTS'  EXPORTING  mode\_zyl\_alerts = 'X'  mandt = sy-mandt  alert\_guid = alert\_guid  x\_alert\_guid = space  \_scope = '3'  \_synchron = space  \_collect = ' '.  ENDMETHOD.  METHOD zyl\_if\_alerting~reprocess\_alert.  alert\_guid = iv\_alert\_guid.  get\_alert( ).  get\_alert\_parameters( ).  CHECK alert\_from\_db\_full IS NOT INITIAL.  CHECK alert\_parameters\_from\_db IS NOT INITIAL.  to\_update\_alert = alert\_from\_db.  to\_update\_alert-reprocess\_counter += 1.  GET TIME STAMP FIELD to\_update\_alert-changed\_on.  to\_update\_alert-changed\_by = sy-uname.  update\_alert( ).  \* CATCH zyl\_cx\_alerting\_error.  \* DATA: lv\_fm\_name TYPE tfdir-funcname,  \* ls\_ptab TYPE abap\_func\_parmbind,  \* lt\_ptab TYPE abap\_func\_parmbind\_tab,  \* ls\_etab TYPE abap\_func\_excpbind,  \* lt\_etab TYPE abap\_func\_excpbind\_tab.  \* FIELD-SYMBOLS: <lv\_value> TYPE any.  DATA(lv\_fm\_name) = alert\_from\_db\_full-funcname.  CHECK lv\_fm\_name IS NOT INITIAL.  SELECT \* FROM fupararef INTO TABLE @DATA(lt\_fupararef) WHERE funcname = @alert\_from\_db\_full-funcname.  IF sy-subrc <> 0.  RETURN.  ENDIF.  \* LOOP AT lt\_fupararef ASSIGNING FIELD-SYMBOL(<ls\_fupararef>).  \* CLEAR: ls\_ptab, ls\_etab.  \* CASE <ls\_fupararef>-paramtype.  \* WHEN 'I'.  \* ls\_ptab-kind = abap\_func\_exporting.  \* WHEN 'E'.  \* ls\_ptab-kind = abap\_func\_importing.  \* WHEN 'T'.  \* ls\_ptab-kind = abap\_func\_tables.  \* WHEN 'I'.  \* ls\_ptab-kind = abap\_func\_changing.  \* WHEN 'X'.  \* ls\_etab-name = <ls\_fupararef>-parameter.  \* ls\_etab-value = <ls\_fupararef>-pposition.  \* INSERT ls\_etab INTO TABLE lt\_etab.  \* CONTINUE.  \* ENDCASE.  \* ls\_ptab-name = <ls\_fupararef>-parameter.  \*  \* CREATE DATA ls\_ptab-value TYPE (<ls\_fupararef>-structure).  \*  \* ASSIGN ls\_ptab-value->\* TO <lv\_value>.  \*  \* READ TABLE alert\_parameters\_from\_db ASSIGNING FIELD-SYMBOL(<ls\_alert\_parameter>) WITH KEY para\_name = <ls\_fupararef>-parameter.  \* IF sy-subrc = 0.  \* <lv\_value> = <ls\_alert\_parameter>-para\_value.  \* ENDIF.  \* INSERT ls\_ptab INTO TABLE lt\_ptab.  \*  \* ENDLOOP.  \*  \* "call function  \* CALL FUNCTION lv\_fm\_name  \* PARAMETER-TABLE lt\_ptab  \* EXCEPTION-TABLE lt\_etab.  CALL FUNCTION 'Z\_YL\_REPROCESS\_ALERT' DESTINATION 'NONE' "IN BACKGROUND TASK AS SEPARATE UNIT  EXPORTING  iv\_func\_name = lv\_fm\_name  it\_alert\_params\_db = alert\_parameters\_from\_db  IMPORTING  et\_return = rt\_bapiret.  ENDMETHOD.  METHOD get\_alert.  SELECT SINGLE \* INTO @alert\_from\_db\_full FROM zyl\_alerts  WHERE alert\_guid = @alert\_guid.  MOVE-CORRESPONDING alert\_from\_db\_full TO alert\_from\_db.  ENDMETHOD.  METHOD get\_alert\_parameters.  SELECT alert\_guid, para\_name, para\_value INTO TABLE @alert\_parameters\_from\_db FROM zyl\_alerts\_param  WHERE alert\_guid = @alert\_guid.  ENDMETHOD.  METHOD zyl\_if\_alerting~complete\_exist\_alert\_from\_cust.  DATA(lt\_existing\_alert) = get\_existing\_open\_alert( is\_alert ).  IF lt\_existing\_alert IS NOT INITIAL.  DATA ls\_alert\_update TYPE zyl\_alerts\_s.  MOVE-CORRESPONDING lt\_existing\_alert[ 1 ] TO ls\_alert\_update.  ls\_alert\_update-alert\_status = '5'.  TRY.  update\_alert( is\_alert\_update = ls\_alert\_update ).  CATCH zyl\_cx\_alerting\_error.  ENDTRY.  ENDIF.  ENDMETHOD.  METHOD get\_existing\_open\_alert.  SELECT \* FROM zyl\_alerts INTO TABLE @rt\_existing\_alert  WHERE alert\_source = @zyl\_if\_alerting=>alert\_source\_custom  AND reftype = @is\_alert-reftype  AND reference = @is\_alert-reference  AND funcname = @is\_alert-funcname  AND alert\_status IN ( 1, 2, 3, 4 ).  ENDMETHOD.  METHOD zyl\_if\_alerting~alert\_from\_custom.  IF ib\_alert\_error = abap\_true.  MOVE-CORRESPONDING is\_alert TO to\_update\_alert .  to\_update\_alert\_parameters = it\_alert\_parameter.  to\_update\_alert-alert\_source = zyl\_if\_alerting=>alert\_source\_custom.  create\_alert( ).  IF alert\_guid IS NOT INITIAL.  create\_parameters\_for\_alert( ).  ENDIF.  rv\_alert\_guid = alert\_guid.  ELSE.  zyl\_if\_alerting~complete\_exist\_alert\_from\_cust( is\_alert = is\_alert ).  ENDIF.  ENDMETHOD.  METHOD map\_bapi\_to\_behv\_message.  CLEAR: ro\_message\_behv.  \*--------------------------------------------------------------------\*  \* Map message  \*--------------------------------------------------------------------\*  CREATE OBJECT ro\_message\_behv TYPE lx\_t100  EXPORTING  severity = me->if\_rap\_plmi\_bapi\_msg\_convert~convert\_msgty\_to\_behv\_severity( iv\_message\_type = is\_message\_bapi-type )  msgid = is\_message\_bapi-id  msgno = is\_message\_bapi-number  msgv1 = is\_message\_bapi-message\_v1  msgv2 = is\_message\_bapi-message\_v2  msgv3 = is\_message\_bapi-message\_v3  msgv4 = is\_message\_bapi-message\_v4.  ENDMETHOD.  METHOD if\_rap\_plmi\_bapi\_msg\_convert~convert\_msgty\_to\_behv\_severity.  CLEAR: rv\_severity\_behv.  CASE iv\_message\_type.  WHEN 'A' OR 'E'.  rv\_severity\_behv = if\_abap\_behv\_message=>severity-error.  WHEN 'W'.  rv\_severity\_behv = if\_abap\_behv\_message=>severity-warning.  WHEN 'I'.  rv\_severity\_behv = if\_abap\_behv\_message=>severity-information.  WHEN 'S'.  rv\_severity\_behv = if\_abap\_behv\_message=>severity-success.  WHEN OTHERS.  " Not valid - set error  rv\_severity\_behv = if\_abap\_behv\_message=>severity-error.  ENDCASE.  ENDMETHOD.  METHOD get\_idoc\_status\_error.  DATA: ls\_status\_error TYPE LINE OF ty\_r\_status\_error.  SELECT teds1~status, teds2~descrp  FROM teds1  LEFT OUTER JOIN teds2 ON teds1~status = teds2~status  INTO TABLE @DATA(lt\_status)  WHERE teds2~langua = 'E'.  IF sy-subrc = 0.  ls\_status\_error-option = 'EQ'.  ls\_status\_error-sign = 'I'.  LOOP AT lt\_status ASSIGNING FIELD-SYMBOL(<fs\_status>).  IF NOT <fs\_status>-descrp CS 'Error' AND  NOT <fs\_status>-descrp CS 'error' AND  NOT <fs\_status>-descrp CS 'not' .  DELETE lt\_status INDEX sy-tabix.  CONTINUE.  ENDIF.  ls\_status\_error-low = <fs\_status>-status.  APPEND ls\_status\_error TO rr\_status\_error.  ENDLOOP.  ENDIF.  ENDMETHOD.  METHOD zyl\_if\_alerting~create\_alert\_messages.  DATA: lt\_zyl\_alerts\_msgs TYPE TABLE OF zyl\_alerts\_msgs,  ls\_zyl\_alerts\_msgs TYPE zyl\_alerts\_msgs.  IF it\_idoc\_msgs IS NOT INITIAL.  LOOP AT it\_idoc\_msgs ASSIGNING FIELD-SYMBOL(<fs\_idoc\_msg>).  ls\_zyl\_alerts\_msgs-alert\_guid = iv\_alert.  ls\_zyl\_alerts\_msgs-msg\_counter = sy-tabix.  ls\_zyl\_alerts\_msgs-msg = <fs\_idoc\_msg>-statxt.  APPEND ls\_zyl\_alerts\_msgs TO lt\_zyl\_alerts\_msgs.  CLEAR ls\_zyl\_alerts\_msgs.  ENDLOOP.  DATA(lv\_lines) = lines( it\_idoc\_msgs ).  ENDIF.  IF it\_bapiret2 IS NOT INITIAL.  LOOP AT it\_bapiret2 ASSIGNING FIELD-SYMBOL(<fs\_bapiret2>).  ls\_zyl\_alerts\_msgs-alert\_guid = iv\_alert.  ls\_zyl\_alerts\_msgs-msg\_counter = sy-tabix + lv\_lines.  CALL FUNCTION 'MESSAGE\_TEXT\_BUILD'  EXPORTING  msgid = <fs\_bapiret2>-id  msgnr = <fs\_bapiret2>-number  msgv1 = <fs\_bapiret2>-message\_v1  msgv2 = <fs\_bapiret2>-message\_v2  msgv3 = <fs\_bapiret2>-message\_v3  msgv4 = <fs\_bapiret2>-message\_v4  IMPORTING  message\_text\_output = ls\_zyl\_alerts\_msgs-msg.  APPEND ls\_zyl\_alerts\_msgs TO lt\_zyl\_alerts\_msgs.  CLEAR ls\_zyl\_alerts\_msgs.  ENDLOOP.  ENDIF.  IF lt\_zyl\_alerts\_msgs IS NOT INITIAL.  MODIFY zyl\_alerts\_msgs FROM TABLE lt\_zyl\_alerts\_msgs.  ENDIF.  ENDMETHOD.  METHOD get\_slg1\_messages.  DATA: lt\_log\_header TYPE balhdr\_t,  lt\_msg\_handle TYPE bal\_t\_msgh,  ls\_msg TYPE bal\_s\_msg,  lt\_msg TYPE bal\_t\_msg,  lt\_bapiret2 TYPE bapiret2\_t,  ls\_bapiret2 TYPE bapiret2.  \*--------------------------------------------------------------------\*  " Create log filter  \*--------------------------------------------------------------------\*  IF iv\_ext\_identif IS NOT INITIAL.  DATA(lt\_bal\_sel\_extn) = VALUE bal\_r\_extn( ( sign = 'I' option = 'EQ' low = iv\_ext\_identif ) ).  ENDIF.  IF iv\_object IS NOT INITIAL.  DATA(lt\_bal\_sel\_obj) = VALUE bal\_r\_obj( ( sign = 'I' option = 'EQ' low = iv\_object ) ).  ENDIF.  IF iv\_subobject IS NOT INITIAL.  DATA(lt\_bal\_sel\_sobj) = VALUE bal\_r\_sub( ( sign = 'I' option = 'EQ' low = iv\_subobject ) ).  ENDIF.  DATA(ls\_log\_filter) = VALUE bal\_s\_lfil( extnumber = lt\_bal\_sel\_extn  object = lt\_bal\_sel\_obj  subobject = lt\_bal\_sel\_sobj ).  \*--------------------------------------------------------------------\*  " Pass the header data and get log header data to find log handle  \*--------------------------------------------------------------------\*  CALL FUNCTION 'BAL\_DB\_SEARCH'  EXPORTING  i\_client = sy-mandt  i\_s\_log\_filter = ls\_log\_filter  IMPORTING  e\_t\_log\_header = lt\_log\_header  EXCEPTIONS  log\_not\_found = 1  no\_filter\_criteria = 2  OTHERS = 3. "#EC CI\_SUBRC  IF sy-subrc = 0.  \*--------------------------------------------------------------------\*  "Get message handler to find actual message  \*--------------------------------------------------------------------\*  CALL FUNCTION 'BAL\_DB\_LOAD'  EXPORTING  i\_t\_log\_header = lt\_log\_header  i\_client = sy-mandt  IMPORTING  e\_t\_msg\_handle = lt\_msg\_handle  EXCEPTIONS  no\_logs\_specified = 1  log\_not\_found = 2  log\_already\_loaded = 3  OTHERS = 4. "#EC CI\_SUBRC  IF sy-subrc = 0.  \*--------------------------------------------------------------------\*  "Get log message and export  \*--------------------------------------------------------------------\*  LOOP AT lt\_msg\_handle ASSIGNING FIELD-SYMBOL(<ls\_msg\_handler>).  CALL FUNCTION 'BAL\_LOG\_MSG\_READ'  EXPORTING  i\_s\_msg\_handle = <ls\_msg\_handler>  i\_langu = sy-langu  IMPORTING  e\_s\_msg = ls\_msg  EXCEPTIONS  log\_not\_found = 1  msg\_not\_found = 2  OTHERS = 3. "#EC CI\_SUBRC  IF sy-subrc = 0.  APPEND ls\_msg TO lt\_msg.  CLEAR ls\_msg.  ENDIF.  ENDLOOP.  IF lt\_msg IS NOT INITIAL.  SORT lt\_msg by time\_stmp DESCENDING.  LOOP AT lt\_msg INTO ls\_msg.  "export message  MESSAGE ID ls\_msg-msgid TYPE ls\_msg-msgty NUMBER ls\_msg-msgno  WITH ls\_msg-msgv1 ls\_msg-msgv2 ls\_msg-msgv3 ls\_msg-msgv4 INTO ls\_bapiret2-message.  ls\_bapiret2-id = ls\_msg-msgid.  ls\_bapiret2-type = ls\_msg-msgty.  ls\_bapiret2-message\_v1 = ls\_msg-msgv1.  ls\_bapiret2-message\_v2 = ls\_msg-msgv2.  ls\_bapiret2-message\_v3 = ls\_msg-msgv3.  ls\_bapiret2-message\_v4 = ls\_msg-msgv4.  ls\_bapiret2-number = ls\_msg-msgno.  APPEND ls\_bapiret2 TO lt\_bapiret2. CLEAR ls\_bapiret2.  ENDLOOP.  SORT lt\_bapiret2.  DELETE ADJACENT DUPLICATES FROM lt\_bapiret2.  ENDIF.  et\_bapiret2[] = lt\_bapiret2[].  ENDIF.  ENDIF.  ENDMETHOD.  ENDCLASS. |

|  |
| --- |
| ZYL\_IF\_ALERTING |
| interface ZYL\_IF\_ALERTING  public .  constants ALERT\_SOURCE\_CUSTOM type ZYL\_ALERT\_SOURCE\_E value '1' ##NO\_TEXT.  constants ALERT\_SOURCE\_IDOC type ZYL\_ALERT\_SOURCE\_E value '2' ##NO\_TEXT.  constants ALERT\_SOURCE\_APOMON type ZYL\_ALERT\_SOURCE\_E value '3' ##NO\_TEXT.  methods ALERT\_FROM\_CUSTOM  importing  !IS\_ALERT type ZYL\_ALERTS\_S\_WITHOUT\_KEY  !IT\_ALERT\_PARAMETER type ZYL\_ALERTS\_PARAM\_T  !IB\_ALERT\_ERROR type ABAP\_BOOLEAN  returning  value(RV\_ALERT\_GUID) type SYSUUID\_C32  raising  ZYL\_CX\_ALERTING\_ERROR .  methods COMPLETE\_EXIST\_ALERT\_FROM\_CUST  importing  !IS\_ALERT type ZYL\_ALERTS\_S\_WITHOUT\_KEY .  methods CREATE\_ALERT\_FROM\_IDOC  importing  !IB\_ALERT\_ERROR type ABAP\_BOOLEAN optional  !IT\_ALERT\_PARAMETER type ZYL\_ALERTS\_PARAM\_T optional  !IS\_ALERT type ZYL\_ALERTS\_S\_WITHOUT\_KEY  returning  value(RV\_ALERT\_GUID) type SYSUUID\_C32  raising  ZYL\_CX\_ALERTING\_ERROR .  methods CREATE\_ALERT\_FROM\_APOMON  importing  !IS\_ALERT type ZYL\_ALERTS\_S\_WITHOUT\_KEY  returning  value(RV\_ALERT\_GUID) type SYSUUID\_C32  raising  ZYL\_CX\_ALERTING\_ERROR .  methods UPDATE\_ALERT  importing  !IS\_ALERT\_UPDATE type ZYL\_ALERTS\_S optional  returning  value(RS\_ALERT) type ZYL\_ALERTS\_S  raising  ZYL\_CX\_ALERTING\_ERROR .  methods SEND\_ALERT  importing  !IV\_ALERT\_GUID type SYSUUID\_C32  returning  value(RB\_SUCCESS) type ABAP\_BOOLEAN  raising  ZYL\_CX\_ALERTING\_ERROR .  methods REPROCESS\_ALERT  importing  !IV\_ALERT\_GUID type SYSUUID\_C32  returning  value(RT\_BAPIRET) type BAPIRET2\_T  raising  ZYL\_CX\_ALERTING\_ERROR .  methods CREATE\_ALERT\_MESSAGES  importing  !IV\_ALERT type ZYL\_ALERTS-ALERT\_GUID  !IT\_IDOC\_MSGS type EDIDS\_TT optional  !IT\_BAPIRET2 type BAPIRET2\_T optional .  endinterface. |

**Functions Modules / RFCs**

|  |
| --- |
| z\_yl\_reprocess\_alert |
| FUNCTION z\_yl\_reprocess\_alert  IMPORTING  VALUE(iv\_func\_name) TYPE rs38l\_fnam  VALUE(it\_alert\_params\_db) TYPE zyl\_alerts\_param\_t  EXPORTING  VALUE(et\_return) TYPE bapiret2\_t.  DATA: ls\_ptab TYPE abap\_func\_parmbind,  lt\_ptab TYPE abap\_func\_parmbind\_tab,  ls\_etab TYPE abap\_func\_excpbind,  lt\_etab TYPE abap\_func\_excpbind\_tab.  FIELD-SYMBOLS: <lv\_value> TYPE any.  SELECT \* FROM fupararef INTO TABLE @DATA(lt\_fupararef) WHERE funcname = @iv\_func\_name.  IF sy-subrc <> 0.  EXIT.  ENDIF.  LOOP AT lt\_fupararef ASSIGNING FIELD-SYMBOL(<ls\_fupararef>).  CLEAR: ls\_ptab, ls\_etab.  CASE <ls\_fupararef>-paramtype.  WHEN 'I'.  ls\_ptab-kind = abap\_func\_exporting.  WHEN 'E'.  ls\_ptab-kind = abap\_func\_importing.  WHEN 'T'.  ls\_ptab-kind = abap\_func\_tables.  WHEN 'I'.  ls\_ptab-kind = abap\_func\_changing.  WHEN 'X'.  ls\_etab-name = <ls\_fupararef>-parameter.  ls\_etab-value = <ls\_fupararef>-pposition.  INSERT ls\_etab INTO TABLE lt\_etab.  CONTINUE.  ENDCASE.  ls\_ptab-name = <ls\_fupararef>-parameter.  CREATE DATA ls\_ptab-value TYPE (<ls\_fupararef>-structure).  ASSIGN ls\_ptab-value->\* TO <lv\_value>.  READ TABLE it\_alert\_params\_db ASSIGNING FIELD-SYMBOL(<ls\_alert\_parameter>) WITH KEY para\_name = <ls\_fupararef>-parameter.  IF sy-subrc = 0.  <lv\_value> = <ls\_alert\_parameter>-para\_value.  ENDIF.  INSERT ls\_ptab INTO TABLE lt\_ptab.  ENDLOOP.  "call function  CALL FUNCTION IV\_FUNC\_NAME  PARAMETER-TABLE lt\_ptab  EXCEPTION-TABLE lt\_etab.  ENDFUNCTION. |

**UI5 Objects**

****